Direct Addressing + Open Addressing

Data Structures of Set ADT

next class: Separate Chaining

Suppose we have a set of n area codes that we want to maintain.

area code, for example, could be one instance var in a "County" object
Direct Addressing

insert, locate, remove

000
001

314 → Object for area code 314 (bucket/collection of all countries with this area code)

999 array ← called a table
Key here

every element you might possibly insert into set has a dedicated index → slot in the table

\[
\text{insert} - \quad \text{table}[\text{slot}] = \text{element} \quad \Theta(1)
\]

\[
\text{locate} - \quad \text{access} \text{table}[\text{slot}] \quad \Theta(1)
\]

\[
\text{remove} - \quad \text{table}[\text{slot}] = = \text{null} \quad \Theta(1)
\]
Let's make this a little more general

equivalence tester

define how check for equivalence

hasher

hashCode mapping from object to some integer in \{0, 1, \ldots, m-1\}

size of table
What is the big limitation?

Size of table is as big as Universe of all possible elements that might be inserted.

Consider a universe of 5000 students where SS # is the id #.
Direct addressing is only a reasonable choice (in terms of space usage) when roughly \( n > \frac{|U|}{4} \)

\# elements held in set
We want time efficiency of direct addressing but we can’t waste so much space.

$10^9$ students $\geq$ $SS$ # $\in$ $U$ $\rightarrow$ collision | Must have lots of elements in universe map to the same slot.
Hash Function

function that maps from `hashcode` to \( \{0, \ldots, m-1\} \)

some integer from \( 0, \ldots, m-1 \)

hash table size

Desired property – each element \( x \in \bigcup_{i=0}^{m} \text{hash}(x, \text{hashcode}(i)) \) is equally likely to be any int in \( \{0, \ldots, m-1\} \)

For \( \text{prob} \frac{1}{m} \)
Pick a hash function to be a mathematical function that maps a random integer to $0, \ldots, m-1$.

One thought: $\text{hashCode mod } m\%$

Pick something that breaks natural patterns that occur.

Multiplication method - multiply by irrational #
Still we can have collisions—what do we do?

Open Addressing

Separate Chaining

these data structures differ in how collisions are handled
hash function

$h(314) \rightarrow \{C_1, C_2, C_3\} \rightarrow h(314) \rightarrow h(636)$

$C_1, C_2, C_3$ area code 314 Mapping Bucket
Separate Chaining

One solution: keep a list of all elements (buckets) that hash to the same slot.
Open Addressing

If the slot you hash to is already occupied (there's a collision), go somewhere else.

Requires there's \( \geq \) one slot per element. \((m \geq n)\)
How do we decide where to go next?

Important that element e always follows the same sequence of slots as it looks for an open one.

probe sequence